An Intelligent Tutoring System for Medical Imaging

Brent Martin[1], Tracy Kirkbride[2], Antonija Mitrovic[1], Jay Holland[1] and Konstantin Zakharov[3]

[1]Intelligent Computer Tutoring Group, University of Canterbury, Christchurch New Zealand
{brent.martin@, tanja.mitrovic@, jah130@student.}canterbury.ac.nz
[2]School of Medical Imaging, Christchurch Polytechnic of Information Technology, Christchurch New Zealand
KirkbrideT@cpit.ac.nz
[3]School of Computing, University of Leeds, United Kingdom
mnkz@leeds.ac.uk

Abstract: Intelligent Tutoring Systems (ITS) are increasingly breaking free of the lab and entering the classroom. These practice-based software systems promise significant learning gains over more traditional e-learning approaches. However, their widespread uptake is still hampered by the difficulty in building them. Over the past ten years we have developed ITS authoring tools that have evolved to the point that it is now feasible for teachers to build their own systems. We report on one such system, an ITS for medical imaging, built by the head of the Medical Imaging Department of a polytechnic institution using VIPER, an ITS authoring system we have developed. Despite requiring no specialist skills (other than domain knowledge) to build, an initial study provides evidence that the resulting ITS is effective at teaching the important concepts of this subject, and demonstrates that is it feasible for teachers to develop their own intelligent tutoring systems.

Introduction

Intelligent Tutoring Systems (ITS) increasingly show promise as a technology that will expand the horizons of education from those able to attend a bricks-and-mortar institution to anyone with an Internet connection. Acting as an enhancement to traditional distance learning offerings, they promise to augment laboratories and tutorials by allowing students to practice the skills they are learning from home. In recent years tutors such as the Geometry and Algebra tutors (Koedinger, Anderson et al. 1997), and the Addison-Wesley database place suite (SQL-Tutor, ER-Tutor and NORMIT) (Mitrovic, Martin et al. 2007) have made it out of the lab and into the classroom. Compared to more traditional e-learning techniques ITS have two major advantages: they separate the domain knowledge from the problem set (thus allowing the same domain knowledge and diagnostic reasoning to be used over and over again), and they tailor the learning experience to individual students by modeling their learning behavior. The former is achieved by building a model of the subject being learned (the "domain model"), and using it to diagnose the student's solution and provide fine-grained feedback. This is then augmented by a "student model" that maps the student's knowledge onto the domain model. A common method is the "overlay" model, in which the system tracks the student's performance for each knowledge unit in the domain model and annotates it with the level to which the student is believed to have learned that concept. Other, more complex approaches also exist (Holt, Dubs et al. 1994).

Intelligent tutoring systems are complex, requiring specialist skills and techniques. Even some of the more popular approaches for domain and student modeling such as model tracing (Anderson, Corbett et al. 1995) have a high overhead, requiring as much as ten hours of development per diagnostic rule (Koedinger, Anderson et al. 1997), and ratios as high as 200 hours of model development for every hour of instruction have been reported (Woolf and Cunningham 1987). Constraint-Based Modeling (CBM) (Ohlsson 1994) is a relatively recent approach for building ITS that supports the building of domain and student models, and which requires less effort than approaches such as model tracing; results as low as 1.1 hours per rule or "constraint" have been reported (Mitrovic, Koedinger et al. 2003). CBM is based on the theory of learning from performance errors (Ohlsson 1996). It models the domain as a set of state constraints, where each constraint represents a declarative concept that must be learned and internalized before the student can achieve mastery of the domain. Constraints represent restrictions on solution *states*, and take the form:

If <*relevance condition*> *is true for the student's solution,*
THEN <*satisfaction condition*> *must also be true*

The relevance condition of each constraint checks whether the student's solution is in a pedagogically significant state. If so, the satisfaction condition is checked. If it succeeds, no action is taken; if it fails, the student has made a mistake, and appropriate feedback is given. *Syntactic* constraints check that the solution is syntactically correct. Conversely, *semantic* constraints check whether the student's solution has solved the problem, usually by comparing it to an "ideal" solution supplied by the teacher. The constraints implicitly encode semantics by testing for all of the different possible encodings of the concept they are attempting to test. The student is thus permitted to use a different problem-solving strategy to the author, or even to mix strategies, provided no fundamental domain concepts are violated. This feature makes CBM very powerful in domains where there is no obvious single "right" answer.

Constraint-based tutors are effective: for example, students using SQL-Tutor have shown significant gains in learning after as little as two hours of exposure to this system (Mitrovic and Ohlsson 1999). CBM reduces the authoring effort (compared to approaches like model tracing) by requiring the author model only states, rather than solution paths (Mitrovic, Koedinger et al. 2003). Nevertheless, the task of building an ITS is still large and complex. To reduce the authoring effort we have developed a number of tools, including WETAS (Martin and Mitrovic 2002) and ASPIRE (Mitrovic, Suraweera et al. 2006), the latter being an authoring system that allows the complete development of an ITS without ever writing a line of code. In 2008 we further extended ASPIRE by building as much of the modeling task (which was previously carried out by the ITS author) into the tool itself, further reducing the complexity of the task of creating the domain model. This new system, called VIPER (Virtual Interactive Practice Environment Resource), was then used by the second author to build an ITS for Medical Imaging.

The Medical Imaging ITS

The Medical Imaging ITS is an environment for practicing various skills related to the field of medical imaging. It consists of a set of domains, each of which contains sets of problems for one or more related tasks. Figure 1 shows the interface for one of the tasks, namely identifying parts of medical equipment. In this example the task is to label various components of the rotating x-ray diagram. Having logged into the system and selected the domain (or "module") of "Medical Equipment", the student then selects the problem set corresponding to the task they wish to

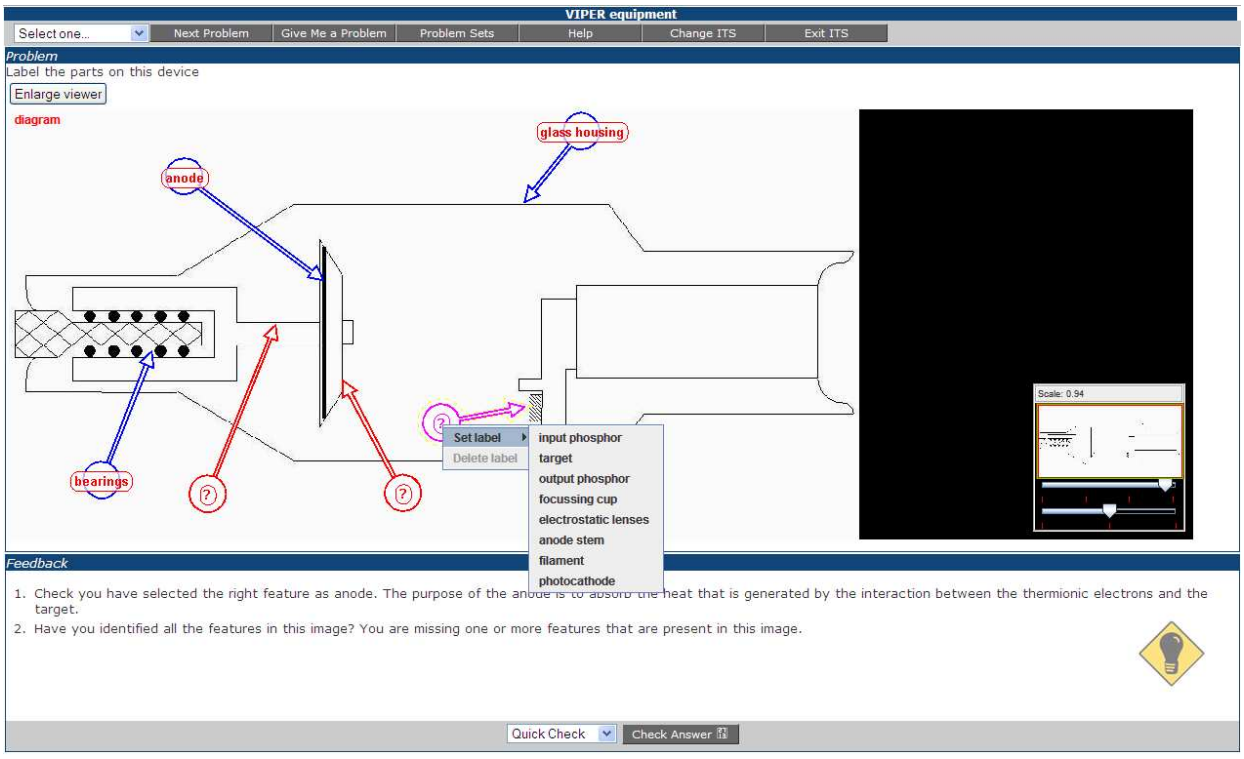

Figure 1: Screenshot of the tutoring system

carry out; in this case there are two: “label an image” (in which the student must supply labels for parts of the diagram) and “identify features in an image” (where they are required to find and click on various parts). They can then select a problem, either by choosing from a list or by asking the system to select the problem it thinks is most suited to their current knowledge of this domain. This latter function is driven by the student model, and is primarily based on the concepts the system believes the student is still struggling to learn. They then click on each arrow and select the label that correctly identifies what the arrow is pointing to; in Figure 1 the student is about to select the label for the filament.

At any point the student may ask for their solution to be diagnosed, and the system will give them feedback on their answer. They may choose the level of feedback presented from the following: “quick check” (indicates whether the answer is correct or not), “Hint” or “Detailed hint” (provides a hint for a single error), “All errors” (lists hints for all mistakes) and “Show solution” (displays the correct solution). In the screenshot the student had selected “All errors” and has been given feedback about their two mistakes: they have labeled the wrong feature as “Anode”, and they have not completed the problem yet (i.e. some labels are still blank). The feedback given is specific to the concept they have mis-used (i.e. identification of the anode), *not* to the specific problem they are solving; any problem requiring identification of an anode will generate the same feedback when this concept is violated. Once the student has completed the problem they may select another problem, change the problem set (i.e. select a different task from the same domain) or change domains.

The Medical Imaging ITS contains a total of six domains: general anatomy, atomic physics, x-ray imaging, imaging equipment, chest anatomy and MRI settings. Within each domain there are one or more tasks spanning image labeling, identifying (clicking on) features in an image, critiquing images, experimenting with the parameters of an image and answering general questions. There are over 300 individual problems in the system to date, with many more currently being authored, representing tens of hours of instruction.

Authoring with VIPER

The VIPER ITS development system consists of two parts: an authoring tool and a tutoring engine: the former is used by the author to develop an ITS in a particular domain, whilst the latter serves up the ITS to the students. In VIPER the authoring of a domain consists of the following main steps:

1. Create a new domain instance, giving it a name and specifying what *type* of domain it is;
2. Create the domain model
3. Enter the set of problems and their solutions
4. Deploy the domain to the tutoring engine

To create a new domain instance the author clicks “Add domain”, enters the domain’s name, and selects which type of domain this is. VIPER reduces the modeling effort required by providing templates for certain types of domain, where any domain of this type will share the same basic domain model structure, diagnostic reason logic and interface type(s). For the medical imaging project we created three domain types, each of which may have one or more interface associated with them, as follows (interface types are given in parentheses):

1. Image analysis (“compare images” and “experiment with parameters”)
2. Image contents (“identify features” and “label image”)
3. Answer a question (“general question” and “fill in the blanks”)

Whilst these domain types were created for the Medical Imaging ITS, they are in no way specific to this domain and in practice could be used for a wide of range of subjects. Each is now described.

“Image analysis” is concerned with the comparison of images and/or the parameters of the image itself. The “compare images” interface requires the student to study two images and select one based on features of the image and to indicate which features led to their conclusion. An example domain that uses this interface is “compare X-ray images”. The student is required to determine which of two images was taken under certain conditions (e.g. which was taken with a higher kVp) and support their decision by indicating what image features they find that support their choice (e.g. darker soft tissue, more anatomical detail). The interface includes an image viewer that enables the student to pan and zoom the two images. For the other interface (“experiment with parameters”), the student is shown an image for which they can modify some parameter and view the results. They are then asked questions

about the effect. An example task we created using this interface is “Experiment with x-ray images”, where the student selects various values for kVp and views the effect on an x-ray image. They then have to indicate what effect increasing kVp has on the image, by selecting from a set of choices.

Whereas image analysis is concerned with the image as a whole, “image contents” is used for tasks where the student is required to identify what is *in* the image. The first interface (“identify features”) requires the student to click on each of a list of features that are present in the image, whilst for the second they are given an image with arrows pointing to various features, for which they must select the correct label. We created domains of this type for medical equipment diagrams and anatomy.

“Answer a question” is a very general domain type that allows the author to build an arbitrary model of concepts that will be required to solve the problem. The two interfaces are simply two different ways of presenting the same kind of problem: in “general question” the student can select multi-choice answers and/or type in free-form text for one or more question sub-parts, while for “fill in the blanks” the answer is provided as a sentence with missing terms, which the student fills in, again either by selecting the correct value or typing the answer. An example of a domain we created of this type is “atomic physics”. For the “general question” interface students calculate half-lives and activity levels by selecting the correct formula from a list, specifying the answer’s units (again by choosing from a list) and typing in the answer. Tables of decay constants, half-lives and formulae are given to the student as appropriate. For “fill in the blanks” students we create the domain of “Ionization and excitation”, where students are shown an energy level diagram for an atom and asked to specify the energy change for various cases of electron movement. The answer took the form:

[{At least | Exactly}] [energy change] eV is [{released | absorbed}]

where {A | B} indicates the student had to choose A or B from a drop-down box, compared to [energy change] which was typed in directly.

The next step is the largest: creating the domain model. The domain model consists of a set of concepts that make up the domain (possibly structured into a hierarchy), together with feedback to be given when the student makes an error associated with that concept. Figure 2 is a screenshot showing a domain model being entered. For each domain type the type of concepts the model is made up of is restricted to a certain vocabulary. For example, in “image content”, the model consists of *features*, i.e. objects that may be seen in the image. Entering the domain model for this domain type therefore consists of entering a list of features that may appear in an image for this

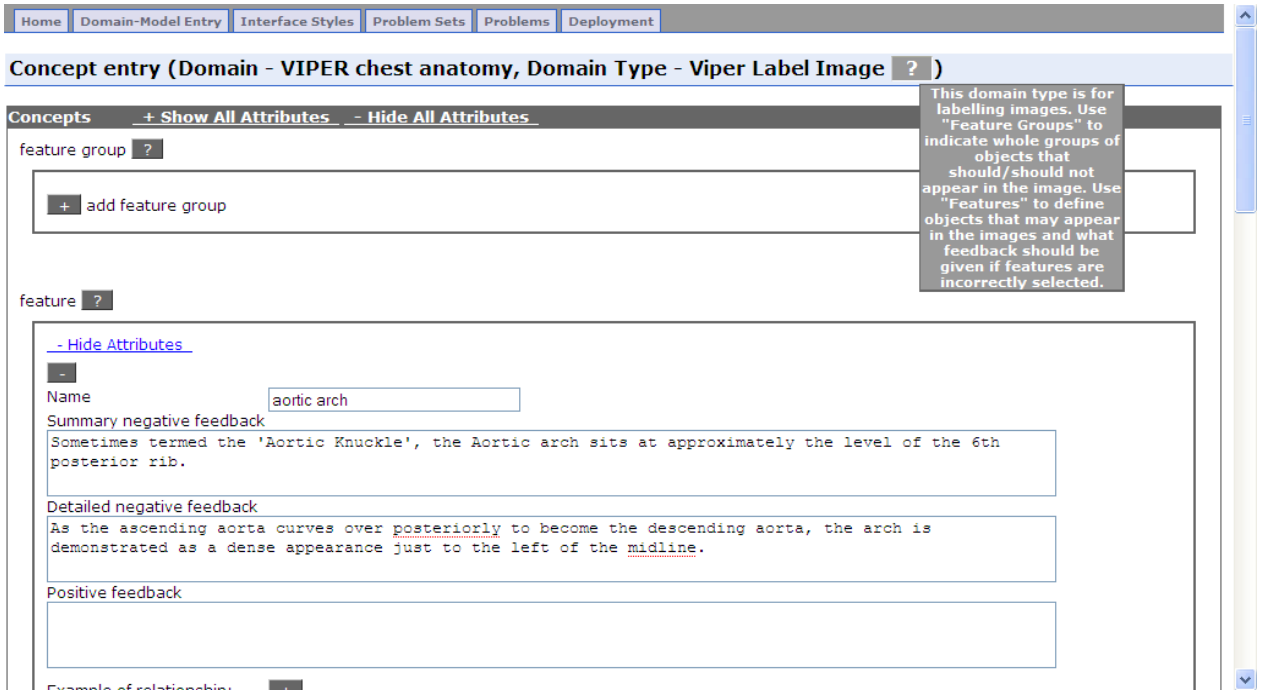

Figure 2: VIPER domain entry interface

domain, and entering the feedback that will be given to the student when each feature is incorrectly used. Note that, depending on the domain type, more than one feedback message may be required for each concept: in "image content" a different message may be delivered when the student has failed to find a feature than when they have labeled the wrong object as being this feature. Features may also be made hierarchical; for example, the author can create the features "colon", "ascending colon" and "descending colon" and arrange then such that the latter two are child features of "colon". This enables specific feedback to be given in the case where the answer is "ascending colon" but the student has specified either "descending colon" or just "colon", for example. Other domain types may have more complex vocabularies. For example, in "Image analysis the student must not only identify the features of an image that are relevant (e.g. "soft tissue", "anatomical detail" for x-ray images), they must also provide a value for the feature: for "soft tissue", the associated values are "darker soft tissue" and "lighter soft tissue". The following is part of the domain model for x-ray analysis (abridged). In this domain type, the feature values may also be related to another concept: in the example below "more anatomical detail" indicates the use of a "low contrast technique".

Feature: anatomical detail
Hint feedback: What is the role of anatomical detail in this question?
Feedback if wrongly used: Is anatomical detail relevant to this question?

Feature value: more anatomical detail
Summary feedback: Having more anatomical detail means that we are using a low contrast technique (i.e. a higher kVp).
Example of: low contrast technique

Feature value: less anatomical detail
Summary feedback: Having less anatomical detail means that we are using a high contrast technique (i.e. a lower kVp).
Example of: high contrast technique

Feature: contrast technique ABSTRACT

Feature value: low contrast technique
Detailed feedback: Using a low contrast technique provides more greys in the image (i.e. a long grey scale); this enables us to visualise anatomy such as trabeculae bone.

The domain model is entered via a form where the author can add new items, fill in the feedback and specify relationships (such as "example of") between a new item and an existing one. Once the domain model has been entered VIPER automatically generates the diagnostic rules that will be used to evaluate a student's solution and provide feedback specific to the student's error(s).

The author then enters problems and their solutions. After adding a new problem and specifying the problem text, the author fills in an example solution using an interface very similar to that used by the student; the only difference is that for tasks such as "label image" the author can create new labels as well as selecting the correct answer for them. Finally, the author selects "deploy"; if there are no problems with the information they have entered the domain is sent to the tutoring engine where it is made available to the students.

Preliminary Evaluation

VIPER has been used by Year 1 and 2 Medical Imaging students at Christchurch Polytechnic of Information Technology (CPIT). A total of 59 students participated in the study, which involved letting them use the system as much or as little as they liked over a period of 2 weeks. The aim of this preliminary study was to gain feedback on the students' impressions of the ITS, and, more importantly, to determine whether or not the diagnosis method and feedback were effective in teaching students in a range of domains. We report on the four most heavily used domains. Table 1 gives some information about usage of the system for these domains.

Domain	Students
Radioactive decay	24
Ionization and excitation	29
Imaging equipment	10
Anatomy	35

Table 1. Summary of student participation

To assess feedback efficacy we plotted "learning curves" for each of the domains served up by VIPER, namely radioactive decay, imaging equipment, ionization and excitation and anatomy. These domains spanned several different tasks as described in the previous section. Learning curves plot the probability of the student making an error when applying a knowledge unit, as a function of the number of opportunities they have had to exercise that knowledge unit, averaged over all knowledge units (constraints in VIPER's case) and over all participating students. Curves are cut off at the point where the number of participating data points (i.e. the number of distinct student-knowledge unit pairs) falls below 25 percent of the initial participation at X=1. If the student is learning the knowledge units the curve will exhibit an inverse power law of decay. Conversely, if the knowledge units are not being learned, the data will appear random. Whilst it is possible that students will learn knowledge units even if we don't explicitly teach them the underlying concepts (i.e. they will simply learn from practice), we expect the feedback VIPER provides to significantly improve learning performance. If this is the case, we would expect the curves for domains that contain feedback to be smoother and/or steeper power laws than those for domains where no feedback is given. The study did not have a formal control group; instead, one domain (anatomy) was provided that did not have any specific feedback; students were simply given a canned message such as "Are you sure you have identified the spleen correctly?"

As well as the lack of feedback for the anatomy domain, there were other differences between the domains, notably the number of knowledge units (constraints), the number of problems, and the level of student participation. Table 2 lists the key parameters for the domains; this information is necessary to interpret the learning curves correctly. "Constraints" indicates the number of constraints that were relevant to at least one submitted solution; "initial data points" indicates the number of data points (i.e. constraint/student pairs) that were relevant for X=1; this gives an indication of how much data was aggregated to produce the curve.

Domain	Constraints	Problems	Initial data points
Radioactive decay	28	207	411
Ionization and excitation	10	65	283
Imaging equipment	63	3	261
Anatomy	552	34	3618

Table 2. Experiment parameters

Figure 3 shows the learning curves for the four domains on which the students practiced the most. The first two are for domains that provided feedback, where there were a large number of problems to solve and high student participation; these domains exhibit very good power law fits ($R^2 > 0.75$) and a similar rate of decay. For the "equipment" domain whilst the number of constraints is high, participation is less than for the first two domains. More importantly, there are fewer problems, and thus fewer opportunities to practice each knowledge unit. Nonetheless, the data still shows a reasonable fit to a power law, and the error is generally decreasing, suggesting learning is taking place. (Note that learning curves degrade rapidly as the data set size reduces.) The final curve is for the Anatomy domain, which has a large number of constraints and had the highest student participation. However, no concept-specific feedback was provided for this domain: it merely told them when they had mis-labelled an anatomical feature. The learning curve for this domain appears random, indicating no significant learning took place during the student sessions. This result must be treated with some caution however; the "drop-off" rate of participation with respect to each knowledge unit is high for this domain because there are a large number of constraints, each of which is relevant only to a small number of problems. However, the Equipment domain had even fewer problems per knowledge unit, yet still exhibited a fairly good power law.

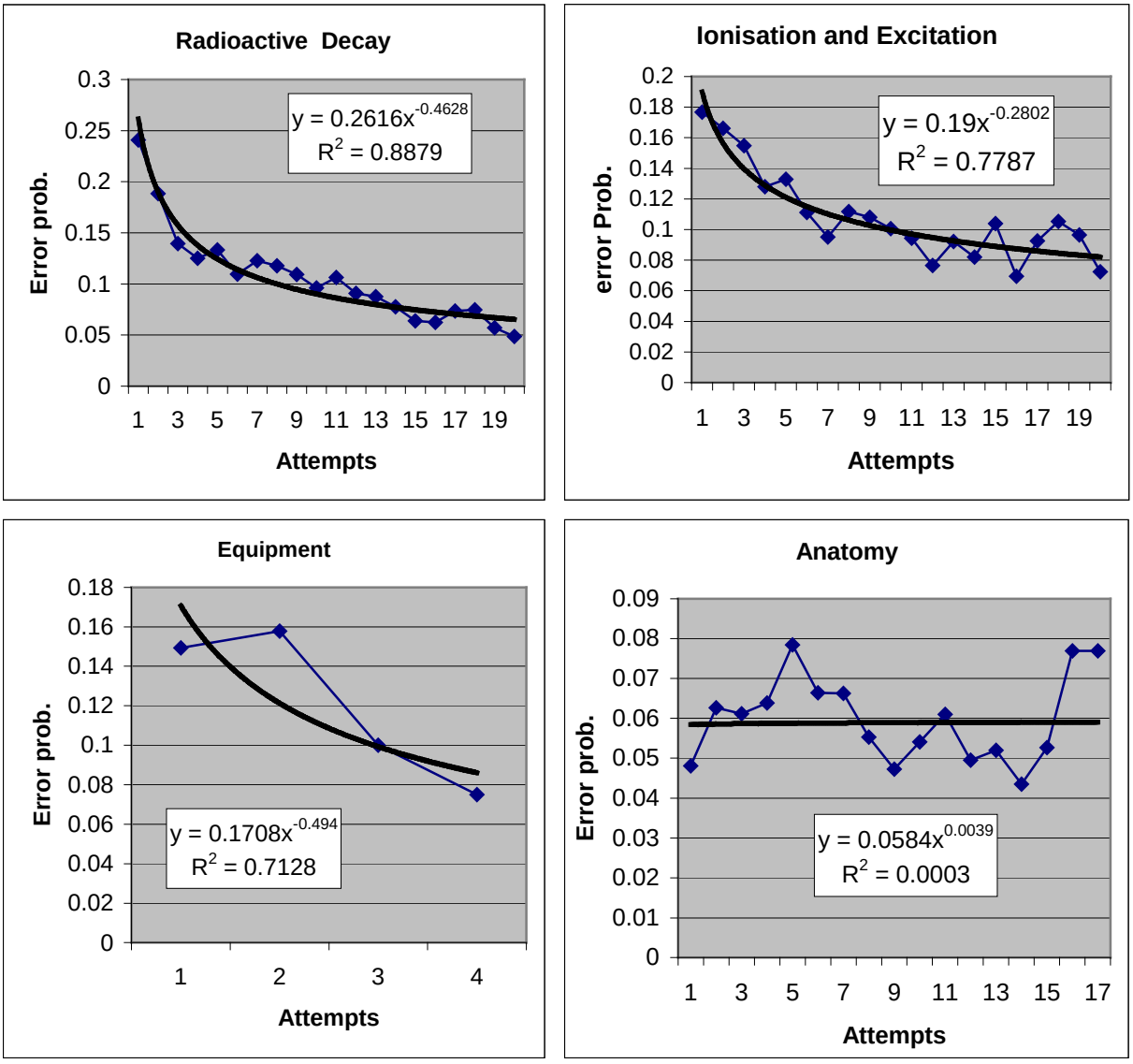

Figure 3. Learning curves for the four most-used domains

Discussion

The Medical Imaging tutor is a fairly simple example of an ITS. Nonetheless to build it from scratch would have required a major development effort, for which the payoff might not have justified the effort. By using VIPER the author's task has been reduced by many orders of magnitude, with the tool performing most of the hard work of building the diagnostic model, and the author effectively just providing content for this model in a way that was not particularly onerous. Nonetheless, the feedback provided as a result of this semi-automated model appears to have been successful in aiding students learning, as evidenced by the learning curves. Further, the system was very positively received by students, so might be expected to have motivational benefits, particularly for "drier" areas of the topic such as atomic physics.

The domain types created for this tutor can easily be applied to other domains: "Image analysis" is applicable to any domain where students would be expected to discriminate between two images, including medical diagnosis, art critiquing and architecture; "image content" can be used in any domain where a students might be expected to identify parts of something that can be presented pictorially, such as electronic equipment, vehicle repairs, paleontology (find the bones) and even less obvious domains such as software design; "general question" (as

its name implies) is a very general domain type with potential to be applied to practically any domain. We are also actively adding new domain types, including "languages", which uses the grammar of one or more languages to support activities such as writing (e.g. computer programs) and translation.

Conclusions

ITS authoring is a difficult task. Whilst generic authoring tools such as ASPIRE dramatically reduce the domain authoring effort required, it nevertheless remains a specialized task. Efforts to ease this burden by restricting the tool to a particular domain type come at the expense of generality. We developed VIPER, a development tool that makes it possible for teachers to use their domain knowledge to create intelligent tutors without the need for specialist skills, and demonstrated its use by building an ITS for medical imaging. An evaluation of the Medical Imaging tutor showed that the feedback it generated was effective in teaching the student the key concepts of the domain. VIPER is general enough to be used in many other domains, and we are continually extending the set of domain types.

Intelligent tutoring systems are a promising tool for delivering education electronically. To date a key problem has been the effort required to build such systems, even when sophisticated authoring tools are used. VIPER is a promising step towards making ITS a realistic option for education practitioners everywhere.

References

Anderson, J. R., A. T. Corbett, et al. (1995). *Cognitive Tutors: Lessons Learned*. Journal of the Learning Sciences 4(2): 167-207.

Holt, P., S. Dubs, et al. (1994). *The State of Student Modeling.* Student Modeling: The Key to Individualized Knowledge-Based Instruction. J. Greer and G. McCalla. New York, Springer-Verlag: 3-39.

Koedinger, K. R., J. R. Anderson, et al. (1997). *Intelligent Tutoring Goes To School in the Big City.* International Journal of Artificial Intelligence in Education 8: 30-43.

Martin, B. and A. Mitrovic (2002). *WETAS: A Web-Based Authoring System for Constraint-Based ITS.* Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Malaga, Springer.

Mitrovic, A., K. R. Koedinger, et al. (2003). *A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling.* Ninth International Conference on User Modeling UM 2003, Springer-Verlag.

Mitrovic, A., B. Martin, et al. (2007). *Intelligent Tutors for All: The Constraint-Based Approach.* IEEE Intelligent Systems July/August 2007: 38-45.

Mitrovic, A. and S. Ohlsson (1999). *Evaluation of a Constraint-Based Tutor for a Database Language*. International Journal of Artificial Intelligence in Education 10: 238-256.

Mitrovic, A., P. Suraweera, et al. (2006). *Authoring constraint-based tutors in ASPIRE*. ITS 2006, Taiwan.

Ohlsson, S. (1994). *Constraint-Based Student Modeling.* Student Modeling: The Key to Individualized Knowledge-Based Instruction. J. Greer and G. McCalla. New York, Springer-Verlag: 167-189.

Ohlsson, S. (1996). *Learning from Performance Errors.* Psychological Review 3(2): 241-262.

Woolf, B. and P. A. Cunningham (1987). *Multiple Knowledge Sources in Intelligent Teaching Systems.* IEEE Expert 2(1): 41-54.